This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A method for executing a transaction comprising at least one query language statement and at least one file system statement, each statement relating to a user defined type ("UDT") associated with a database server, the method comprising:

storing at least one field relating to the UDT in a relational database table, wherein at least one field is a filestream field, and wherein data for each filestream field is stored in a respective file separate from the relational database table;

receiving each of the at least one file system statements, wherein each statement comprises a call to open [[an]] a first item and at least one of a call to read from the first item and to write to the first item, and a call to close the first item~~, wherein each item referenced in a statement is stored separately from a database table associated with the item~~;

receiving each of the at least one query language statement, wherein each query language statement is associated with [[an]] a second item;

for each file system statement:

upon detecting a storage platform path name associated with the first item in a file system statement forwarding the file system statement to an agent, wherein the agent performs a call to the storage platform by passing the storage platform path name to the storage platform;

identifying the first item based upon the storage platform path name;

passing a database engine function that returns a file system path name corresponding to the first item to the database server;

performing a table look-up for the UDT associated with the first item on the database server;

extracting a real file system path corresponding to the first item using the database engine function;

using the real file system path to perform an operation on the first item by passing the real file system path back to the agent, wherein the agent then interacts with the file system to cause execution of file system statement, wherein

if the file system statement includes open, read and close operations:

creating a transaction as part of an open operation, wherein the transaction is managed separately from the database server;

obtaining a read lock on a data table row associated with the associated first item for the file system statement;

performing a read operation in the context of the transaction;

committing the transaction as part of a close operation;

if the file system statement includes open, write and close operations:

creating a transaction as part of an open operation, wherein the transaction is managed separately from the database server;

obtaining a write lock on a data table row associated with the associated first item for the file system statement;

performing a write operation in the context of the transaction;

committing the transaction as part of a close operation;

for each query language statement, starting a transaction on the database server updating fields associated with the second item in the query language statement and sending an updategram to the database server.

2. (Previously Presented) The method of claim 1, wherein the data table row includes a user defined type corresponding to the item.

3. (Previously Presented) The method of claim 1, wherein each query language statement is a T-SQL statement.

4. (Previously Presented) The method of claim 1, wherein transactions created for file system statements are managed by a storage platform.

5. (Previously Presented) The method of claim 1, further comprising receiving a transaction context for file system operations and performing at least one of a read lock and a write lock consistent with the received transaction context.

6. (Previously Presented) The method of claim 1, wherein creating the transaction comprises:

determining whether creating the transaction will result in a conflict;

if creating the transaction will result in a conflict, then resolving the conflict according to a conflict resolution scheme; and

if creating the transaction will not result in a conflict, then starting the transaction.

7. (Original) The method of claim 1, wherein acquiring the read lock on the row comprises acquiring a read committed view of the row.

8. (Original) The method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another transaction from accessing the row while the transaction is being processed.

9. (Original) The method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent a non-transacted file system statement from accessing the row while the transaction is being processed.

10. (Original) The method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another statement within the transaction from writing to the row.

11. (Original) The method of claim 1, wherein acquiring the write lock on the row comprises acquiring a write lock that will enable another statement within the transaction to read from the row.

12. (Original) The method of claim 1, comprising starting the transaction by acquiring one of a read lock and a write lock on a filestream field of the row.

13. (Canceled)

14. (Currently Amended) A method for locking and isolation of a file system statement, the method comprising:

receiving the file system statement comprising a call to open an item, a call to read from the item, and a call to close the item, the file system statement being independent of any database commands employing a query language of a database and wherein each item referenced in the statement is stored separately from a database table associated with the item;

upon detecting a storage platform path name associated with the item in the file system statement, performing a call to a storage platform by passing the storage platform path name to the storage platform;

identifying the item based upon the storage platform path name;

passing a database engine function that returns a file system path name corresponding to the item to a database server;

performing a table look-up for a user defined type associated with the item on the database server;

extracting a real file system path corresponding to the item using the database engine function;

using the real file system path to perform an operation on the item, wherein:

in response to receiving the file system statement that is independent of any database application programming interface requests, determining if a read lock is available for a row of a data table corresponding to the item;

if the read lock is not available for the row of the data table corresponding to the item, then failing the open; and

if the read lock is available for the row of the data table corresponding to the item:

performing a shared open for the item;

acquiring the read lock on the row.

15. (Previously Presented) The method of claim 14, comprising determining if the read lock is available for a row of a data table that includes a user defined type corresponding to the item.

16. (Original) The method of claim 14, wherein acquiring the read lock on the row comprises acquiring a read committed view of the row.

17. (Previously Presented) The method of claim 14, comprising acquiring the read lock on a filestream field of the row.

18. (Canceled)

19. (Currently Amended) A method for locking and isolation of a file system statement, the method comprising:

receiving the file system statement comprising a call to open an item, a call to write to the item, and a call to close the item, the file system statement being independent of any database commands employing a query language of a database and wherein each item referenced in the statement is stored separately from a database table associated with the item;

upon detecting a storage platform path name associated with the item in the file system statement, performing a call to a storage platform by passing the storage platform path name to the storage platform;

identifying the item based upon the storage platform path name;

passing a database engine function that returns a file system path name corresponding to the item to a database server;

performing a table look-up for a user defined type associated with the item on the database server;

extracting a real file system path corresponding to the item using the database engine function;

using the real file system path to perform an operation on the item, wherein:

in response to receiving the file system statement that is independent of any database application programming interface requests, determining if a write lock is available for a row of a data table corresponding to the item;

if the write lock is not available for the row of the data table corresponding to the item, then failing the open; and

if the write lock is available for the row of the data table corresponding to the item:

performing an exclusive open for the item;

acquiring the write lock on the row.

20. (Previously Presented) The method of claim 19, comprising determining if the write lock is available for a row of a data table that includes a user defined type corresponding to the item.

21. (Original) The method of claim 19, wherein acquiring the write lock on the row comprises acquiring a write lock that will prevent another statement from accessing the row while the statement is being processed.

22. (Previously Presented) The method of claim 19, comprising starting the transaction by acquiring the write lock on a filestream field of the row.

23. (Canceled)

24. (Currently Amended) A system for executing a file system statement, the system comprising:

a processor;

a relational data engine comprising a data table having a row corresponding to the item;

a storage platform built on the relational data engine, the storage platform comprising means for receiving the file system statement, wherein each item referenced in the statement is stored separately from a database table associated with the item, means for associating the file system statement with the transaction, and means for starting the transaction in response to receiving the file system statement by acquiring either a read lock or a write lock on the row, the file system statement comprising a call to open an item, a call to read from the item

or to write to the item, and a call to close the item, the file system statement being independent of any database commands employing a query language of a database;

a file system, wherein the file system is adapted to:

upon detecting a storage platform path name associated with the item in the file system statement, performing a call to the storage platform by passing the storage platform path name;

identifying the item based upon the storage platform path name;

passing a database engine function that returns a file system path name corresponding to the item to the relational data engine;

wherein the relational data engine is adapted to:

upon receiving the database engine function from the file system, perform a table look-up for an associated user defined type;

extract a real file system path corresponding to the item using the database engine function;

pass the real file system path to perform an operation on the item to the file system to cause the file system to perform the operation on the item.

25. (Original) The system of claim 24, wherein the row corresponding to the item includes a user defined type corresponding to the item.

26. (Original) The system of claim 24, wherein the storage platform further comprises means for associating a second statement with the transaction.

27. (Original) The system of claim 26, wherein the second statement is another file system statement.

28. (Original) The system of claim 26, wherein the second statement is a transactional query language statement.

29. (Previously Presented) The system of claim 24, wherein the means for starting the transaction comprises means for performing the following steps:

determining whether starting the transaction will result in a conflict;

if starting the transaction will result in a conflict, then resolving the conflict according to a conflict resolution scheme; and

if starting the transaction will not result in a conflict, then starting the transaction.

30. (Original) The system of claim 24, wherein the read lock provides a read committed view of the row.

31. (Original) The system of claim 24, wherein the write lock prevents another transaction from accessing the row while the transaction is being processed.

32. (Original) The system of claim 24, wherein the write lock prevents a non-transacted file system statement from accessing the row while the transaction is being processed.

33. (Original) The system of claim 24, wherein the write lock prevents another statement within the transaction from writing to the row.

34. (Original) The system of claim 24, wherein the write lock enables another statement within the transaction to read from the row.

35. (Original) The system of claim 24, wherein the row comprises a filestream field.